Chapter Four

MATERIALS AND METHODOLOGY

This chapter will outline and discuss the various phases of the research process of the study. It includes information starting from the design process of the study, to building the simulation models, optimizing the parameters, gathering relevant data, measuring the performance, and finalizing the proposed model.

4.1 Design Process

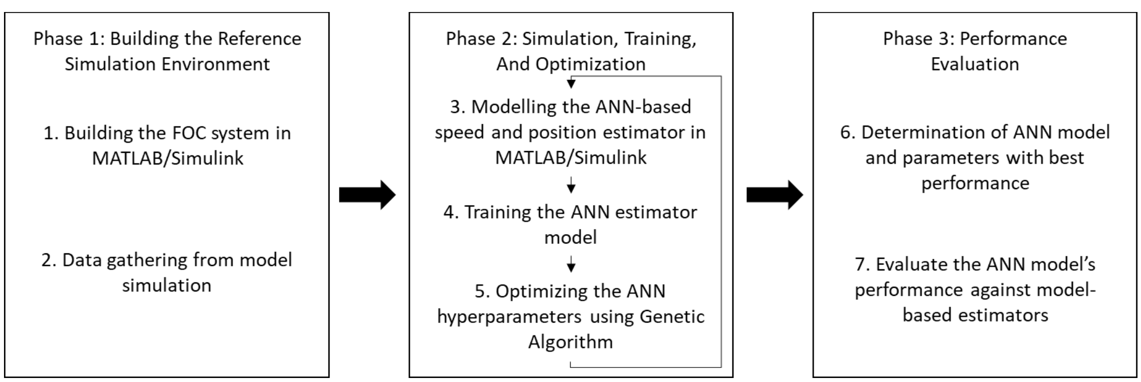

Figure 4.1 Design Process Overview

The design process as shown in Figure 4.1, outlines the three phases of the study. This serves as the overview of the study's methodology. Phase 1 covers *building the reference simulation environment*. Here, the simulation is built, modelled, and configured, basically providing the simulation framework and data set to be used in the next phase. Phase 2 covers the *simulation, training, and optimization* of the speed and position estimator. Using the simulation and data from the previous phase, the process of modelling, training, and optimizing the estimator is conducted here. The last phase is the *performance evaluation* wherein the best performing model is determined and evaluated against a reference model.

4.2 Building the FOC System in MATLAB/Simulink

All simulations in this study are conducted in the MATLAB/Simulink Environment, therefore it is necessary to build there a working field oriented controlled PMSM system. The reference system to be built is the sensorless FOC block diagram in Figure 3.2.

4.2.1 Simulation Workspace

The simulation workspace comprises of the MATLAB interface shown in Figure 4.2 which provides access to a command console and an editor for MATLAB's .m files. The parameters and functions to be used by the simulation can be declared and initialized here as well as in the command console.

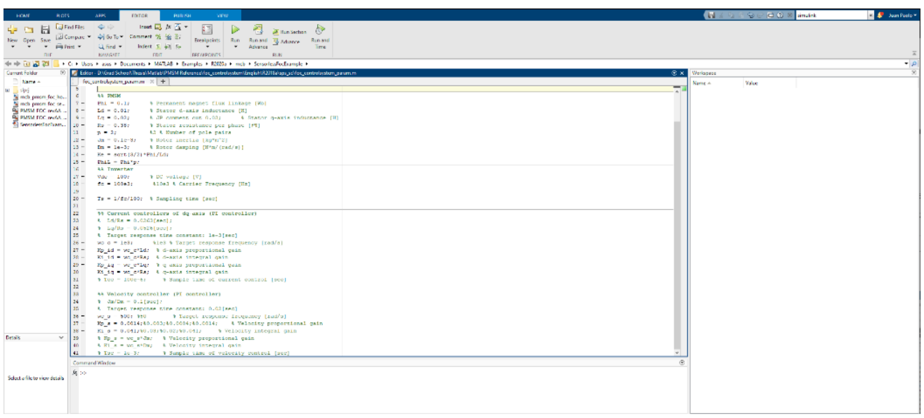

Figure 4.2 MATLAB Workspace

The Simulink environment as shown in Figure 4.3 is integrated with MATLAB that enables the use or incorporation of the parameters, functions, algorithms, among others. Model-based designs are easily realized in this environment especially with the use of the library containing toolboxes and customizable blocks. In the remaining sections of the methodology, relevant Simulink libraries are indicated.

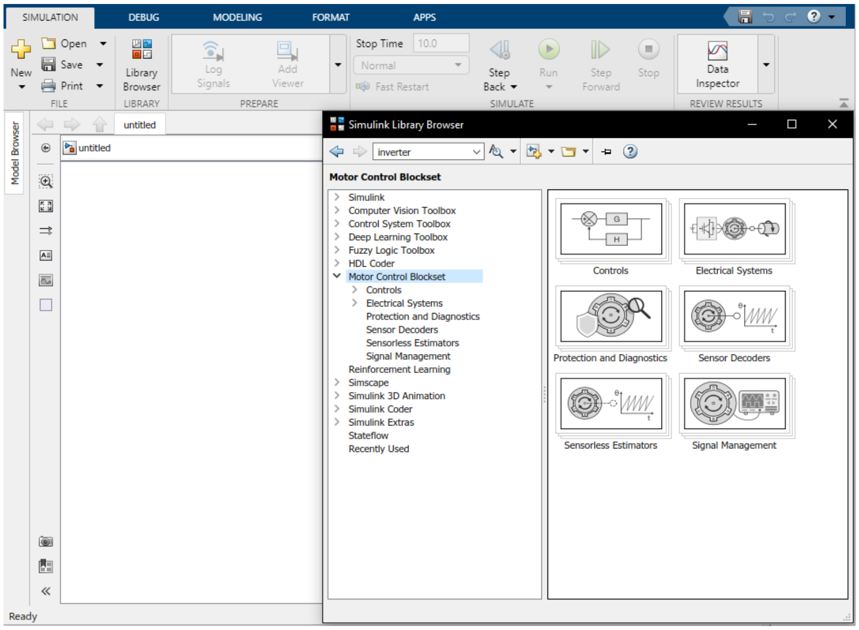

Figure 4.3 Simulink Workspace

4.2.2 Permanent Magnet Synchronous Motor Block

The equivalent permanent magnet synchronous motor block to be used is shown in Figure 4.4. Terminals *a*, *b*, and *c* are the three-phase inputs, *n* is the neutral terminal, *R* is the motor's rotor terminal, and *C* is the motor's case terminal.

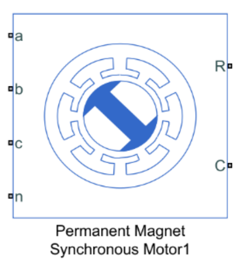

Figure 4.4 Simulink Model of Permanent Magnet Synchronous Motor

The following motor parameters are required in the PMSM block:

1. Winding Type (Wye-wound or Delta-wound)
2. Number of pole pairs
3. Permanent Magnet flux linkage (Wb)

4. Stator d- and q-axis inductances
5. Stator resistance per phase

The motor parameters from the X2212 980KV II Motor are used as the reference in this study:

Table 4.1 Motor Parameters

Parameter	Value
Rated Power	0.3 kW
Rated Speed	9800 rpm
Rated Torque	0.125 Nm
Pole pairs	7
Stator-winding resistance	0.39 Ω
Stator-winding inductance	12.1 mH
Flux Linkage	0.1 Wb
Inertia	0.001 kg m^2

4.2.3 Mechanical Load

Figure 4.5 represents the motor's mechanical load. The mechanical load is represented by an inertia block, a rotational damper, and an ideal torque source, shown in the figure from left to right. The inertia and damping coefficient are to be established The step block is to be configured with the load's torque.

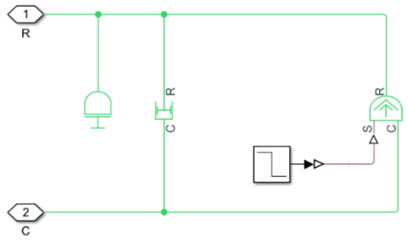

Figure 4.5 Simulink Mechanical Load

The following mechanical parameters are to be established:

1. Inertia $(kg \cdot m^2)$
2. Damping Coefficient $(N \cdot m/(\frac{rad}{s})$

4.2.4 Direct-Quadrature-Zero Blocks

The Clarke and Park transforms, collectively known as the direct-quadrature-zero transforms, including their respective inverse blocks are represented in Simulink with the library blocks shown in Figure 4.6. These blocks will perform the operation as indicated in Section 3.1.4.

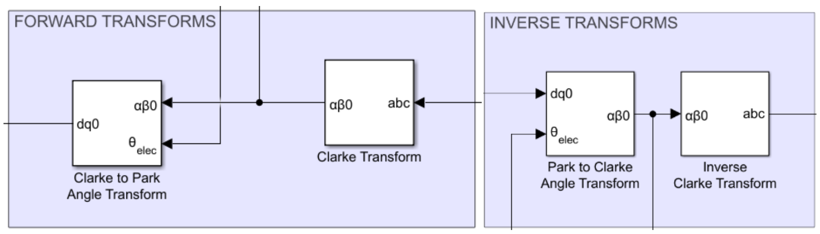

Figure 4.6 Park and Clarke Transforms Simulink Blocks

4.2.5 Sensor Peripherals

The simulation model needs to be prepared for the data-gathering process as well as the validation of performance. Sensor peripherals are also introduced in the model to gather relevant information. The speed and position of the motor are the critical information required by the model and the training data. These measurements are obtained by the motion sensor block shown in Figure 4.7

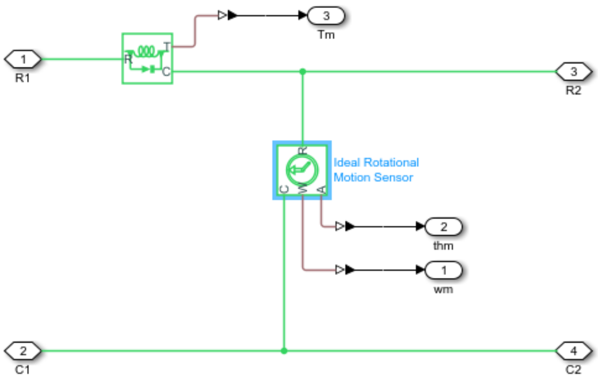

Figure 4.7 Sensor Peripherals

where *wm* represents the rotor speed, *thm* represents the rotor position, and for the qualification stages, *Tm* is the torque of the motor. This will provide the reference speed and position values in the dataset.

4.3 Data Gathering from Model Simulation

Upon building the simulation model in MATLAB/Simulink, data gathering will proceed to establish a data set of input and target values for the neural network training. For data gathering, values for motor parameters such as the stator resistance and the stator inductances should be established. The loading conditions of the motor to be simulated are established as well. Similar to most datasets, data preprocessing needs to be conducted as well prior to training.

4.3.1 Establish Motor Parameters and Loading Conditions

In building up the dataset of the simulation, the test conditions during the simulation run should be established first. Motor parameters are to be established based on a nominal value as well as the values indicative of the parameter's tolerances. These values are to be based on a reference datasheet for a PMSM. The dataset should also consider different motor loads during steady-state and dynamic loading conditions.

4.3.2 Features Selection

In the field of machine learning, features are considered as individual properties to be observed, given that they are measurable [11]. The quality of the features has a great impact on the performance of the network. The features serve as the inputs of our neural network which are then mapped to a target output. Consequently, these features along with the target output comprises the data set.

Given that the proposed neural network estimator is to replace the sliding mode observer, the features selected for the neural networks are the following:

1. Stator voltages: V_α, V_β
2. Stator currents: I_α, I_β

During simulation of the reference FOC, the above features are measured and recorded along with the corresponding measured position of the rotor, θ, and the sine and cosine of θ.

4.3.3 Data Preprocessing

To ensure the quality of the data set used to train the neural network, the data set is to undergo a preprocessing. The workflow for the preprocessing of data is shown in Figure 4.8.

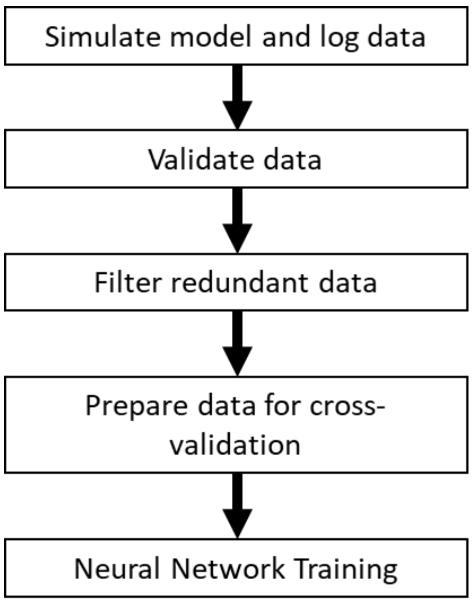

Figure 4.8 Data Preprocessing Workflow

After simulation and logging the data, the next step is to v*alidate the data*. This validation is done by means of plotting the data gathered and comparing the waveforms obtained from the simulation's oscilloscopes with the plotted data. Mismatches will be easy to identify using this method to capture any mismatch, missing values, and outliers that may have been introduced when the data is handled. After, there is a need to *filter redundant data*. For an FOC-based system in steady-state for a certain set of motor parameter values and loading condition, the data gathered should be periodic, as such, the data per such condition is limited to one electric period. Once verified and filtered, the data is to be *prepared for cross-validation.* A split approach is to be considered with the following distribution: Training data (70%), Validation data (15%), and Testing data (15%).

4.4 Modelling the ANN-based Speed and Position Estimator in MATLAB/Simulink

The neural network estimator is built in the MATLAB/Simulink Environment. This is achieved with the help of the *Deep Learning Toolbox* from MATLAB. Certain information required in performing this stage are dependent on the Genetic Algorithm to be discussed in section 4.6, this includes the randomized initial hyperparameters.

4.4.1 Simulink Model

In the Simulink model for the estimator, the neural network will be represented by a user-defined function as shown in Figure 4.9. The user-defined function will represent the neural network model with all of its weights and biases with considerations on the input and outputs. The model will consider the neural network as a function during simulation of the FOC.

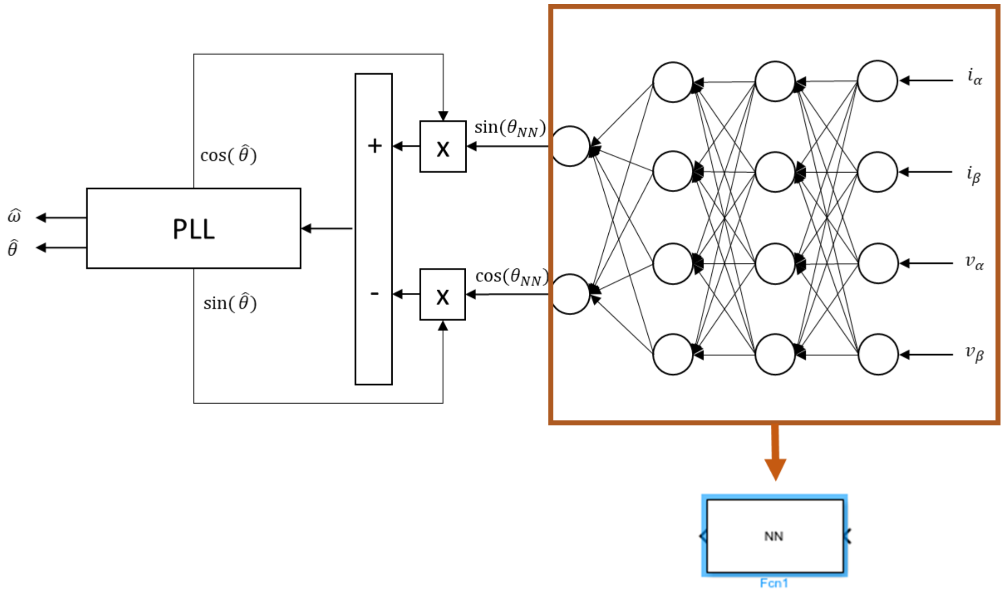

Figure 4.9 Equivalent Simulink Neural Network Block

4.4.2 Neural Network Configuration

The Deep Learning Toolbox provides a graphical user interface for configuring and training neural networks; however, it still offers limited flexibility. This design will make use of command-line functions for modelling the network, setting its hyperparameters, and training on the given data set.

The feed-forward neural network is considered for this study. Initial and succeeding hyperparameters are determined by the genetic algorithm discussed in section 4.6.

4.5 Training the ANN Estimator Model

Once the neural network has been configured, the training will proceed. In training the neural network, the Deep Learning Toolbox will still be used. Command-line functions that are declared in the script to be used enables the user to set the configurations for training. In this section, the training process and algorithm are discussed.

In training the neural network, the training algorithm used has a considerable impact on the ability of the network to learn. The training algorithm determines how to adjust the weights and biases of the network based on the cost function. There are already pre-configured training algorithms in the Deep Learning Toolbox and can be called in the training configurations of the neural network. The training algorithm is a property of the network object and can easily be set to a training function.

For this study, the training algorithm to be used is the *Levenberg-Marquardt* Back-propagation algorithm. This algorithm is selected as it offers flexibility between simpler functions and more complex ones.

4.6 Optimizing the ANN Hyperparameters using Genetic Algorithm

As stated in this study, previous research involving the use of neural network estimators in field oriented control of PMSMs make use of trial and error or using the previous experimental data in determining certain hyperparameters. This study will introduce the use of genetic algorithm in optimizing the network hyperparameters for this application.

The process for genetic algorithm to be used in this study is shown in Figure 4.10. As discussed in Chapter 3, the algorithm will start off with initializing a population. Each individual of the population will have its fitness evaluated. From the data of fitness values, the termination criteria is checked if already met. If not, the population enters parent selection wherein individuals are paired based on their individual fitness to generate an offspring. This offspring is developed by the crossover operator. The mutation operator is also used to help avoid convergence on a local minima. This will ensure the effectiveness of the optimization process.

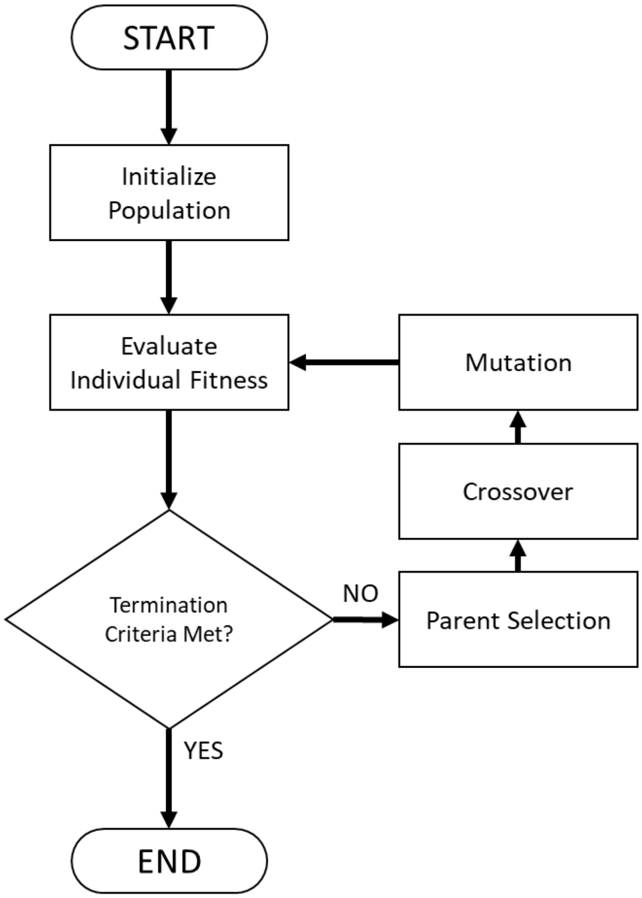

Figure 4.10 Genetic Algorithm Workflow

In developing the genetic algorithm for the speed and position estimator, the termination criteria should be set. These criteria can come in the form of a maximum number of generations and/or in the form of a target mean-squared error (MSE).

As for the hyperparameters to be optimized, a valid range of values should be declared from which the genetic algorithm will limit itself to. This range can be realized during the linear mapping process. The initial population is used as the initial values for the hyperparameters in the neural network design. The target hyperparameters to be optimized are the following:

Table 4.2 Hyperparameters for Optimization

Hyperparameter	**Description**
Number of hidden layers	Determines the number of hidden layers in the neural network. A minimum and maximum number of hidden layers are to be set.
Number of neurons for the hidden layers	Determines the number of neurons for the different hidden layers of the neural network. A minimum and maximum number of neurons are also set.
Activation Function	The activation function for the neural network can be configured as well. (Log-sigmoid, tangent-sigmoid, and linear)
Batch size	Configures the training batch size.
Initial Mu	Initial value for the adaptation parameter in Levenberg-Marquardt, represented by μ in Equation 3.20
Mu decrease factor	The rate at which the adaptation parameter is decreased during successful steps (improvement in cost function)
Mu increase factor	The rate at which the adaptation parameter is increased during unsuccessful steps (deterioration in cost function)

The genetic algorithm stops when either a predetermined number of generations has been reached or when the fitness value of an individual has met a set value. The fitness function used is the mean-squared error.

4.7 Determination of ANN Model and Parameters with Best Performance

After optimizing multiple configurations of neural networks, the model with the best performance will be the proposed estimator. In this stage of the design process, the relevant data especially on the performance of the network are tabulated and compared. In determining the model with the *best performance,* the following factors are considered:

- Estimation Accuracy
- Network Complexity

The neural network estimators' accuracy is the prime factor in considering the best performing model. Since the accuracy directly affects the performance of the field-oriented control scheme. The model yielding the lowest mean-square error after optimization is considered first. However, the network complexity is also considered. Network complexity is also given consideration, should different models produce similar results, a network with fewer neurons and/or layers would be considered as more efficient in terms of computational requirements.

4.8 Evaluate the ANN Model's Performance Against Conventional Estimators

The evaluation stage provides the data on the performance of the proposed GA-optimized neural network estimator against conventional estimators such as the SMO and the MRAS as well as two reference neural network estimators. The data will provide an insight into the relative strengths and weaknesses of the estimators in different conditions.

4.8.1 Load Selection

The qualification of the FOC system is done on an arbitrary load of 0.8 Nm. In determining the representative mechanical load that the motor could support, the equivalent force is computed using Equation 4.231, and the equivalent mass is taken using Equation 4.21312.

$$F = \frac{\tau}{l} \tag{4.1}$$

Where F is the equivalent force, τ is the load torque in Nm, and *l* is the lever arm.

$$m = \frac{F}{g} \tag{4.2}$$

Where m is the object's mass, F is the calculated force, and g is the gravitational constant.

To consider the thrust required during take-off, the calculated mass was divided by 2 to serve as a conservative margin. In doing the process mentioned, the equivalent mass that the motor could support at 0.8Nm load is 349g. This value would represent a load of 1.396 kg on a quadcopter.

4.8.2 Steady-state Estimation Accuracy

In evaluating the proposed GA-NN estimator, the estimation of the motor speed and position is determined during steady state conditions. At 4000 rpm and 0.8 Nm load, the performance of the GA-NN estimation is compared vs the actual speed and position and the estimated values of the other reference models. The peak-to-peak and the average errors for the speed and position estimation of the motor are to be measured. The performance of the estimation is compared with the other estimator models.

4.8.3 Speed Range Estimation Accuracy

Determining the performance of the estimators over a speed range gives an insight if the estimators have flexibility in terms of the target speed during estimation. A ramp input for the reference or the target speed is used to obtain the estimation accuracy of the models over a speed range. The range set is from 1000 rpm to 9000 rpm. This condition is to be conducted on the proposed GA-NN against the reference estimators.

The ramp input would increase at a rate of 1000 rpm per 0.1 sec and will be taken at a load of 0.8 Nm.

4.8.4 Step Response Performance

The step response provides several indicators on the performance of a given control loop for a dynamic input. The reference or the target speed of the system is set from 0 rpm to 4000 rpm at 0.8 Nm load. The FOC system will use the estimated speed and position of the estimator models instead of the values retrieved from the sensors. The rise time, settling time, overshoot, and the steady-state error are measured for each system. These parameters will provide an insight on the dynamic performance of the estimators.

4.8.5 UAV Flight Plan Performance

Using a reference UAV Flight simulation, the reference speeds for certain flight plan conditions are exported. The flight plan will consist of the target speeds given by the controller over the course of a given flight condition (take-off, climb, descend, and hover). This flight plan will then serve as the reference speed of the FOC system.

The flight plan is generated by using the Simulink project [86]. The target elevation is being altered by a flight plan that would demonstrate the take-off, hover, climb, and descend response of the UAV. The generated target motor speeds by the UAV controller is exported and is used as the input reference speed by the proposed GA-optimized neural network estimator. The reference speed is compared against the actual speed when using the estimator models at 0.8 Nm load.

4.8.6 Efficiency Performance

The efficiency of the FOC system given the estimator model is determined by measuring the power drawn from the DC Supply that is acting as the battery of the system. The reference speed is set at 4000 rpm and the load is set at 0.8 Nm. The input power is taken at steady-state condition by averaging the product of the battery voltage and the current drawn by the converter. The Simulink simulation circuit for retrieving the input power is shown in Figure 4.11. The efficiency is compared between the estimators.

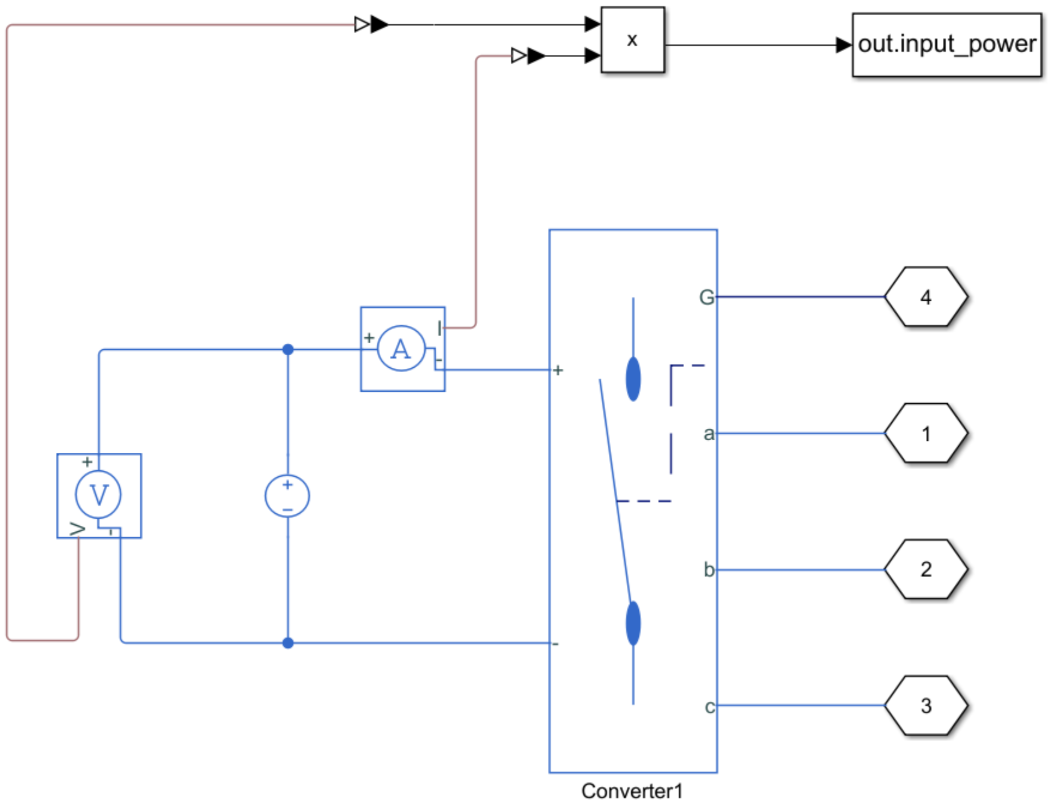

Figure 4.11 Simulink Efficiency Circuit

4.9 Equipment to be Used

For this study, the following equipment and tools are to be used:

1. Laptop

The laptop to be used for the simulation should be capable of the computing requirements of the simulation software. The laptop hardware will determine the total training, optimization, and testing time required for all models and test conditions. Table 4.3 describes the specifications of the laptop used in these simulations.

Table 4.3 Laptop Specifications

Specification	Description
CPU	AMD Ryzen 5 3550H
GPU	AMD Radeon^{TM} Vega 8
RAM	12 GB
Storage	256GB SSD / 1TB HDD
Operating System	Windows 10 Home 64-bit 10.0

2. Simulation Software

The simulation software to be used is the MATLAB R2021A with Simu